(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 169 565**
A2

(12) # EUROPEAN PATENT APPLICATION

(21) Application number: 85109333.6

(22) Date of filing: 25.07.85

(51) Int. Cl.⁴: G 06 F 9/30
G 06 F 9/44

(30) Priority: 25.07.84 JP 154824/84

(43) Date of publication of application:
29.01.86 Bulletin 86/5

(84) Designated Contracting States:
DE FR GB IT NL SE

(71) Applicant: NEC CORPORATION
33-1, Shiba 5-chome, Minato-ku
Tokyo 108(JP)

(72) Inventor: Katori, Shigetatsu
c/o NEC Corporation 33-1, Shiba 5-chome
Minato-ku Tokyo(JP)

(72) Inventor: Maehashi, Yukio
c/o NEC Corporation 33-1, Shiba 5-chome
Minato-ku Tokyo(JP)

(74) Representative: Glawe, Delfs, Moll & Partner
Patentanwälte
Postfach 26 01 62 Liebherrstrasse 20
D-8000 München 26(DE)

(54) **Microprocessor compatible with any software represented by different types of instruction formats.**

(57) A microprocessor include a central processing unit which executes a program according to at least one control signal generated by an instruction decoder. The instruction decoder is designed such that a first type insruction adaptable for the central processing unit can be decoded. A second type instruction unadaptable for the central processing unit is applied as an address to a conversion memory in which the first type instruction corresponding to the second type instruction in their functions is preliminarily stored. The first type instruction in the conversion memory is applied to the instruction decoder instead of the second type instruction. Thus, the second type instruction can be executed by the central processing unit unadaptable for the second type instruction.

EP 0 169 565 A2

./...

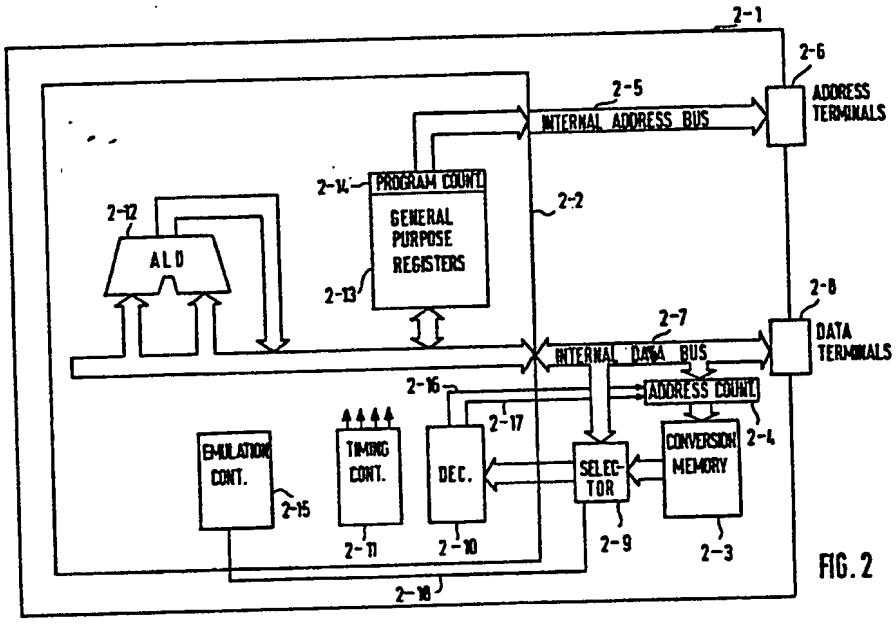

2-1
2-6
ADDRESS TERMINALS
2-5
INTERNAL ADDRESS BUS
2-14
PROGRAM COUNT
GENERAL PURPOSE REGISTERS
2-2
2-12
ALU
2-13
2-8
DATA TERMINALS
2-7
INTERNAL DATA BUS
2-16
ADDRESS COUNT
2-4
2-17
EMULATION CONT.
2-15
TIMING CONT.
DEC.
SELECTOR
CONVERSION MEMORY
2-11
2-10
2-9
2-3
2-18

FIG. 2

pw

MICROPROCESSOR COMPATIBLE WITH ANY SOFTWARE REPRESENTED BY DIFFERENT TYPES OF INSTRUCTION FORMATS

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to a microprocessor including an instruction decoder for decoding instructions to be executed, and more particularly to a microprocessor compatible with softwares which are represented by different types of instruction formats.

Description of the Prior Art

In comformity with the rapid progress of LSI techniques, the efficiency of a microprocessor (or a microcomputer) has remarkably improved. Recently, a microprocessor with a high performance is offered at a low cost. Further, in response to urgent requirements, new types of microprocessors implementing architectures designed on the basis of the latest LSI technology have been developed and been widely used in many industrial fields. Therefore, development of a software adaptable for a newly designed microprocessor is required at a short period of time.

On the other hand, turning eyes on a hardware architecture of a microprocessor, 16-bit or 32-bit parallel processing

type of microprocessor has been developed. On this situation a large number of softwares developed for 8-bit parallel processing type of microprocessor can not be directly adapted to the 16-bit or 32-bit parallel processing type microprocessor (hereinafter, referred to as 16-bit or 32-bit microprocessor). Therefore, new softwares or modified softwares are required for the 16-bit or 32-bit microprocessor. It is, however, impossible to develop new softwares or modify the existing softwares because of necessities of a long period, a complex handwork and a high cost.

To avoid this problem in a conventional 16-bit microprocessor system, 8-bit microprocessor is incorporated in the system. That is, both the 16-bit microprocessor and the 8-bit microprocessor are independently installed and employed in the system. In the operation of this system, when the software for the 16-bit microprocessor is to be executed, the 8-bit microprocessor is controlled such that a data bus and an address bus of the 8-bit microprocessor are discoupled from a system data bus and a system address bus to which a memory is coupled. On the other hand, when the 8-bit microprocessor is to be used, the 16-bit processor is discoupled from the system data bus and the system address bus. This decoupling operation is performed by an interface device inserted between the system buses and the 8-bit and 16-bit microprocessors.

According to this system, two microprocessors are independently required. In other words, two central processing units and two instruction decoders are required. Further, the interface device for controlling the bus connection is required. Therefore, the system has a disadvantage that its size becomes very large. Furthermore, the system cost is enhanced because of many hardware elements. Moreover, in the system operation, the instruction decoder and the

central processing unit (CPU) of the 8-bit microprocessor can not be used when an instruction of the 16-bit microprocessor is executed, and vice versa.

SUMMARY OF THE PRESENT INVENTION

An object of the present invention is to provide a microprocessor having a common hardware resource for softwares, each of which is represented by a different instruction format.

Another object of the present invention is to provide a microprocessor adaptable for both of two types of softwares (e.g. softwares of 8-bit microprocessor and softwares of 16-bit microprocessor).

Still another object of the present invention is to provide a microprocessor capable of performing the existing software without renewal or modification.

Still another object of the present invention is to provide a microprocessor in which a simple hardware circuit performing different types of instructions is formed on a single semiconductor chip.

A microprocessor of the present invention includes a central processing unit, an instruction decoder and a conversion memory. The instruction decoder has a decoding function such that a plurality of control signals are generated on the basis of instructions to be executed by the central processing unit. The conversion memory stores a plurality of instructions which can be executed by the central processing unit.

According to the microprocessor of the present invention, an instruction which can be directly executed by the central processing unit is applied to the instruction

decoder. While, an instruction which can not be executed by the central processing unit as it is applied to the conversion memory as an address of the conversion memory. In response thereof, the instruction stored in the conversion memory is read out from the address designated by the instruction which can not be directly executed by the central processing unit, and is applied to the instruction decoder. Namely, the instruction which can not be directly executed by the central processing unit is converted into the instruction which can be adapted to the central processing unit by means of the conversion memory. Therefore, if the instruction of a type which can not be adapted to the central processing unit is applied to the microprocessor, it is automatically changed to the instruction of a type which is adaptable to the central processing unit and thereafter transferred to the instruction decoder. Thus, a single central processing unit and a single instruction decoder can be commonly used for two types of instructions. Further, the interface device for controlling the bus connection can be omitted from the microprocessor of the present invention. In other words, two types of instructions can be performed without changing the bus connection.

For switching the instructions to be applied to the instruction decoder, the microprocessor of the present invention includes means for detecting the types of the instructions and means responsive to the detecting means for selectively applying the instructions from the system data bus and the instruction from the conversion memory to the instruction decoder.

In the microprocessor of the present invention, a table-look-aside memory, a table reference memory or the like can be used as the conversion memory. Further, by providing an address controller described hereinafter, a variety of instruction conversion can be performed.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a system block diagram of a conventional microprocessor system employing 8-bit microprocessor and 16-bit microprocessor;

Fig. 2 is a block diagram of a microprocessor according to one embodiment of the present invention;

Fig. 3 is an address format diagram of an address controller employed in the microprocessor as shown in Fig. 2;

Figs. 4(a), 4(b) and 4(c) illustrate instruction conversion of the microprocessor shown in Fig. 2; and

Fig. 5 is a system block diagram of a microprocessor system using the microprocessor of the present invention.

DESCRIPTION OF THE PRIOR ART

At first a conventional microprocessor system capable of performing a first type software of 16-bit microprocessor and a second type software of 8-bit microprocessor will be described with reference to Fig. 1. A 16-bit microprocessor 1-1 and a 8-bit microprocessor 1-2 are independently employed and are coupled to a system data bus 1-4 and a system address bus 1-5 via an emulation control circuit (an interface device) 1-3. A memory 1-6 including the first type software and the second type software is coupled to the system data bus 1-4 and the system address bus 1-5 to which an external input-output device 1-7 is also coupled. Hereinafter, a mode in which the first type software is executed is called as a native mode, while one in which the second type software is executed is called as an

emulation mode. The first type of software is represented by instruction codes which can be directly executed by the 16-bit microprocessor. In contrast the second type of software is represented by instruction codes which can be directly executed by the 8-bit microprocessor.

Now, when the system executes a program in the native mode, the 16-bit microprocessor is coupled to the system data bus 1-4 and the system address bus 1-5 according to the control of the emulation control circuit 1-3. At this situation, the 8-bit microprocessor 1-2 is discoupled from the system buses 1-4 and 1-5. The 16-bit microprocessor 1-1 performs the first type of software read out of the memory 1-6. On the other hand, when the second type of software is to be executed, the emulation control circuit 1-3 electrically cuts off the 16-bit microprocessor from the system buses 1-4 and 1-5, and couples the 8-bit microprocessor to the system buses. The emulation control circuit 1-3 acts as a multiplexer according to the selected mode.

As described above, the 16-bit microprocessor 1-1 can execute the first type of software but can not execute the second type of software. While the 8-bit microprocessor 1-2 can execute the second type of software but can not execute the first type of software. In other words, instuctions of the first type of software can be decoded an instruction decoder included in the 16-bit microprocessor 1-1 only, while instructions of the second type of software can be decoded an instruction decoder included in the 8-bit microprocessor only. Therefore, both microprocessors 1-1 and 1-2 are independently required to perform the first tpye and second type softwares.

Further, these two microprocessors must be used exclusively. Particularly, if the two types of softwares are alterna-

tively performed, many bus changing operations are required.

Thus, the conventional microprocessor is voluminous and expensive. Further it is inadequate to the system in which different types of instructions are manipulated. Here, different types of instructions include such instructions that the functions are the same but instruction code patterns are different from each other.

DATAILED DESCRIPTION OF AN EMBODIMENT OF THE PRESENT INVENTION

Fig. 2 shows a block diagram of an embodiment of the present invention. A microprocessor 2-1 has a central processing unit (CPU) 2-2, a conversion memory 2-3, an address control circuit 2-4 and an instruction selector 2-9, all of which are integrated on a single semiconductor chip. The CPU 2-2 includes an instruction decoder 2-10, a timing control circuit 2-11, an arithmetic logic unit (ALU) 2-12, a plurality of general purpose registers 2-13, a program counter 2-14 and an emulation control circuit 2-15. The CPU 2-2 is so designed that, for example, instructions for the 16-bit microprocessor are directly executed. That is, the instruction decoder 2-10 may be the same decoder as that of the conventional 16-bit microprocessor except for one function described hereinafter. The timing control circuit 2-11, the ALU 2-12, the general registers 2-13 and the program counter 2-14 may be also the same circuits as those of the conventional 16-bit microprocessor. Operations of those cirucits are substantially the same operations as that of the conventional 16-bit microprocessor, and therefore the detailed description of their operations is omitted.

The program counter 2-14 is coupled to address bus terminals 2-6 via an internal address bus 2-5 and generates addresses for accessing at least one external memory. The ALU 2-12

and the general purpose registers 2-13 are coupled to data bus terminals 2-8 via an internal data bus 2-7 through which instructions and data are transferred. The address bus terminals 2-6 and the data bus terminals 2-8 are coupled to a system address bus and a system data bus (not shown), respectively, to which at least one instruction memory (not shown) storing both the first type software and the second type software is coupled outside the microprocessor 2-1. However, the instruction memory may be incorporated on the chip of the microprocessor.

The program counter 2-14 is incremented by +1 whenever an instruction is executed by the CPU 2-2. If a jump instruction, a branch instruction or the like is executed, a jump address or a branch address is set in the program counter 2-14. A content of the program counter 2-14 is transferred to the memory coupled to the system address bus via the address bus terminals 2-6. Thus an instruction accessed by the content of the program counter 2-14 is transferred to the internal data bus 2-7 via the data terminals 2-8.

Now, if the first type of software which can be executed by the CPU 2-2 is accessed, the instruction on the internal data bus is applied to the instruction decoder as it is through the instruction code selector 2-9 which is controlled by the emulation control circuit 2-15 such that the internal data bus 2-7 is coupled to the instruction decoder 2-10. Thus, the instruction of the first type of software is directly inputted to the instruction decoder 2-10 and is decoded. The CPU 2-2 executes a program according to the result of decoding. This mode is the native mode.

On the other hand, when the emulation mode is required, the emulation control circuit 2-15 applies a control signal indicating the emulation mode to the instruction code

selector 2-9 through a control signal line 2-18. The instruction code selector 2-9 cuts off the internal data bus 2-7 from the instruction decoder 2-10 and coupled the output of the conversion memory 2-3 to the instruction decoder 2-10. In the emulation mode, an instruction accessed by a content of the program counter 2-14 is one of the second type of software which can not be executed by the CPU 2-2. That is, the instructions to be executed in the emulation mode are those which are executed by the 8-bit microprocessor but can not be executed by the 16-bit microprocessor.

In the present invention the instruction of the second type of software is applied to the conversion memory 2-3 as an address. At the location designated by the instruction on the internal data bus 2-7, the instruction which can be directly executed by the CPU 2-2 is preliminarily stored. This instruction is one of the first type of software and has the same function as that of the instruction of the second type of software. That is, an instruction of the second type of software is converted into the instruction of the first type of software and is applied to the instruction decoder 2-10. Therefore, even if the second type of software is accessed by the microprocessor 2-1, the CPU 2-2 can execute operation exactly. As the result, the CPU of the 16-bit microprocessor and the instruction decoder of the 16-bit microprocessor can execute the second type of software adaptable for the 8-bit microprocessor as well as the first type of software.

According to the present invention, a new development or modification of the software is not required. Further, the CPU and the instruction decoder of the 16-bit microprocessor can be used not only the first type of software bus also the second type of software in common. Therefore, a size and a cost can be extremely reduced.

In the above embodiment, relationship of the first type of software and the second type of software is preliminarily recognized by a user or a manufacturing maker. Therefore, the table to be stored in the conversion memory 2-3 is preliminarily determined.

For this reason, a mast ROM, a programmable ROM, a RAM or the like can be used as the conversion memory. For example, a ROM included in a single chip microcomputer may be used as the conversion memory. Further, each of instructions of the second type of software is used as addresses of the conversion memory, respectively, and the respective instruction of the first type of software corresponding to that of the second type of software is stored at the address location designated by the instruction of the second type of software.

Since the emulation control circuit 2-15 is used to indicate the selecting operation of the selector 2-9, the emulation control circuit 2-15 may be constructed by a well known flip-flop circuit in which a first signal level is set when the native mode is required, while a second signal level is set when the emulation mode is required. Moreover, the emulation control circuit 2-15 may not be located inside the CPU 2-2.

In case that the number of byte(s) of an instruction of the second type of software is equal to that of the corresponding instruction of the first type of software, the instruction of the second type of software can be directly applied to the conversion memory 2-3. However, when essential instruction format, e.g. the number of bytes, of the first type of software is different from that of the second type of software, direct access by the instruction of the second type of software is difficult because of a difference of the number of bytes. Here, one byte

means a part of an instruction which can be accessed by one address.

To desolve this problem an address control circuit 2-4 is provided in the embodiment as shown in Fig. 1. The address control circuit 2-4 includes a register or a latch circuit with at least three fields. If a byte of an instruction of the second type of software consists of 8 bits, the register structure of the address control circuit 2-4 is shown in Fig. 3. In Fig. 3, the register has a first field 3-1 (bit 9), a second filed 3-2 (bits 1 to 8) and a third field (bit 0). The first field 3-1 whose input end receives a first signal 2-17 generated from the instruction decoder 2-10 is used to output a page designating signal. The second field 3-2 is coupled to the internal data bus 2-7 and receives an instruction of 8 bits transferred from the external memory storing the second type of software. The instruction set in the second field 3-2 is directly outputted from its output end without modification. The third field 3-3 receives a second signal 2-16 from the instruction decoder 2-10 and outputs a byte designating signal. In this embodiment, the first signal 2-17 is used as the page designating signal and is set in the MSB (most significant bit) of the register, while the second signal 2-16 is used as the byte designating signal and is set in the LSB (least significant bit) of the register:

Next, the principle of the emulation mode will be described with reference to Fig. 4. As mentioned above, in the emulation mode, an instruction received in the microprocessor 2-1 can not be directly executed because of a different instruction format from that of the microprocessor 2-1. Thus, if the instruction is executed without any conversion, meaningless processing is carried out and the program is in the abnormal state. The present invention employs the conversion memory 2-3 and the address control circuit 2-4.

Thus, the instruction for 8-bit microprocessor is translated (interpretted) by means of the conversion memory 2-6 and then transferred to the instruction decoder 2-10.

However, the corresponding relation of the instructions between the first type of software and the second type software is very complicated. For example, identical instructions, such as a transfer instruction, an arithmetic instruction, are used in general in any microprocessors, but their code patterns are different from each other. Therefore, various cases are considered such as the case that certain operation is assigned by a 1-byte instruction in the second type of software and the same operation therewith is assigned by another 1-byte instruction in the first type of software (hereinafter, reffered to as a frist case), the case that the operation assigned by 1-byte instruction in the second type software is assigned by 2-byte instruction in the first type of software (hereinafter, referred to as a second case, the case that the operation assigned by a 2-byte instruction in the second type of software is assigned by a 2-byte instruction in the first type of software (hereinafter, referred to as a third case) and the like.

At first, the first case will be described with reference to Fig. 3(a). According to an address from the program counter 2-14, a 1-byte instruction is inputted to the internal data bus 2-7 through the data bus terminals 2-8. This 1-byte instruction consisting of 8 bits are set in the second field 3-2 of the register of the address control circuit 2-4. At this time, the page designating signal 2-17 and the byte designating signal 2-16 are both absent, so that the MSB (1st field 3-1) and the LSB (2nd field 3-3) holds "0". If the content of the second field is 10000001, the content of the register is 0100000010. Therefore, 0100000010 is applied to the conversion memory

2-3 as the address. The instruction of the first type of software corresponding to the instruction set in the second field 3-2 is preliminarily stored at the address (0100000010) of the conversion memory 2-3. Thus, the converted instruction is read out ot the conversion memory 2-3 and is applied to the instruction decoder 2-10 through the instruction selector 2-9. The CPU 2-2 executes the instruction according to the result of the decoding operation and advances the content of the program counter 2-14 to access the next instruction of the second type of software.

Secondly, the second case will be described with reference to Fig. 3(b). In this case, an instruction of the second type of software to be executed consists of a single byte (1-byte instruction) , so that only a single byte instruction is received into the microprocessor 2-1. This single byte instruction is set in the second section as well as in the first case. At this time, since the page designating signal 2-17 and the byte designating signal 2-16 are both absent, "0" is set in the first and third fields 3-1 and 3-3 of the register. If, therefore, the code of the single byte instruction set in the second field 3-2 is 11101111, the address to be applied to the conversion memory 2-3 is 0111011110. In this embodiment, a first byte of the instruction of the first type of software corresponding to the signle byte instruction received in the microprocessor 2-1 is preliminarily stored at the address (0111011110) of the conversion memory 2-3. Therefore, the first byte is transferred to the instruction decoder 2-10 through the selector 2-9 and is decoded. As the result of the decoding, the byte designating signal 2-16 is generated. To allow this decoding, an information indicating that the instruction to be executed consists of two bytes is included in the first byte. This technique is known in the art, that is, in a microprocessor manipulating an instruction consisting of a plurality of bytes.

According to the byte designating signal 2-16, "1" is set in the third field 3-3 of the register. Therefore, the address to be applied to the conversion memory 2-3 at the next timing is modified to 0111011111 at which a second byte of the instruction of the first type software to be executed is preliminarily set. Thus, the second byte is successively read out of the conversion memory 2-3.

Next, the third case will be described with reference to Fig. 3(c). In this case, a first byte of an instruction of the second type of software is firstly received in the microprocessor 2-1 and is set in the second field 3-2. At this timing the page designating signal 2-17 and the byte designating signal 2-16 are both absent, and therefore the MSB (1st field 3-1) and the LSB (3rd field 3-3) are "0", respectively. Now, when the first byte set in the second field 3-2 is 10101011, an address applied to the conversion memory 2-3 is 0101010110. In this case, a specific code is preliminarily stored at the location corresponding to the address (0101010110). The specific code includes the information indicating that the current instruction of the second type of software is a 2-byte instruction. This specific code is read out of the conversion memory 2-3 and is applied to the instruction decoder 2-10. The instruction decoder 2-10 decodes the specific code and generates the page designating signal 2-17 and an increment signal for incrementing a content of the program counter 2-14 by +1. The page designating signal 2-17 may be used as the increment signal. As the result, the address advanced by +1 is outputted from the microprocessor 2-1 and is applied to the external memory storing the 2-byte instruction in the second type of software. Thus, the second byte of the instruction to be executed is read out of the external memory and is transferred to the internal data bus 2-7. This second byte is set in the second field 3-2. At this timing, "1" is set in the first filed 3-1 by

the page designating signal 2-17. Therefore, if the second byte set in the second field 3-2 is 11101111, a content of the register is 1111011110. Thus, the conversion memory 2-3 is accessed by the address 1111011110. At the address 1111011110 of the conversion memory 2-3 a first byte of the first type of software corresponding to the instruction of the second type of software to be executed is preliminarily stored. The first byte read out of the conversion memory 2-3 includes an information indicating that a second byte subsequent to the read-out first byte is to be read out of the conversion memory 2-3 successively. Therefore, the instruction decoder 2-10 generates the byte designating signal 2-16 by decoding the read-out first byte. As the result, the content of the register in the address control circuit 2-4 becomes 1111011111. Accordingly, at the subsequent timing, the second byte preliminarily stored at the address 1111011111 is read out of the conversion memory 2-3 and is transferred to the instruction decoder 2-10 through the instruction selector 2-9. Thus, the 2-byte instruction of the first type of software can be read out of the conversion memory 2-3 and is executed by the CPU 2-2. In other words, the CPU can execute the 2-byte instruction of the second type of software in the emulation mode.

In this embodiment, the third field controlled by the byte designating signal is positioned at the LSB field to make manipulation of the 2-byte instruction in the first type of software easy. That is, the access of the second byte of the 2-byte instruction in the first type of software can be performed by merely changing the bit of LSB from "0" to "1". In other words, in this operation modification of the first and the second fileds are not required. Therefore, the address control can be simplicated. If the number of bits of the third field is increased, instructions more than two bytes can be accessed on the basis of the same way.

Further, the first field is used to allow the access of the 2-byte instruction in the second type of software. Namely, by changing pages in the conversion memory 2-3 with respect to the instructions each of which has the different number of bytes, the accesses for any number of bytes can be performed. Therefore, if the number of bits in the first field is increased, any instructions consisting of more than two bytes can be easily converted into the instructions which can be directly executed by the CPU 2-2.

Fig. 4 is a system block diagram of an example of a processing system in which the microprocessor is employed. The address bus terminals 2-6 and the data bus terminals 2-8 of the microprocessor 2-1 in Fig. 1 are coupled to the system address bus 1-5 and the system data bus 1-4, respectively. These system buses 1-4 and 1-5 are further coupled to the external memory 1-6 and the input-output device 1-7. The external memory 1-6 includes the first type of software and the second type of software. The microprocessor 2-1 can execute the first type of software in the native mode and also execute the second type of software in the emulation mode as described below.

In the native mode, the emulation control circuit 2-15 sends a first control signal indicating that the instruction decoder 2-10 is to be coupled to the internal data bus 2-7 to the selector 2-9 through the control signal line 2-18. A flip-flop can be used as the emulation control circuit 2-15 and is set or reset the native mode. If the "reset" is used for the native mode, "0" level signal is applied to the selector 2-9 as the first signal. Under this condition, a content of the program counter 2-14 is transferred to the external memory 1-6 via the address bus terminals 2-6 and the system address bus 1-5 for reading an instruction in the first type of software out of the external memory 1-6. The accessed instruction is transferred

to the internal data bus 2-7 via the system data bus 1-4 and the data bus terminals 2-8 and is directly applied to the instruction decoder 2-10 through the selector 2-9. The decoder 2-10 decodes the instruction and generates a plurality of control signals including an increment signal applied to the program counter 2-14 according to a timing signal which is produced by the timing control circuit 2-11. The program counter 2-14 is incremented according to the incrementing signal. The CPU 2-2 executes an operation according to the control signals. When the execution is terminated, the content of the program counter 2-14 is outputted from the address bus terminals in response to a timing signal indicating an instruction fetch operation. Thus, the first type of software is executed without use of conversion memory 2-3 in the native mode. In this mode, the instruction to be executed by the CPU 2-2 can be directly decoded by the instruction decoder 2-10.

On the other hand, in the emulation mode an instruction read out of the external memory 1-6 is one of the second type of software which can not be directly executed by the CPU. In order to switch over the native mode into the emulation mode, the native mode should include such a last instruction that indicates to change the state of the state of the flip-flop of the emulation control circuit 2-15. Thus, if the flip-flop is in the reset state in the native mode, it is set at "1" before the instructions of the emulation mode are applied to the data bus terminals 2-8. The "1" level signal is applied to the selector 2-9 as a second control signal indicating that the instruction decoder 2-10 is to be coupled to the conversion memory 2-3. The set operation of the flip-flop may be controlled by an output of the decoder 2-10 which is generated on the basis of a mode designating instruction included in the first type of software or in response to a mode designating switch.

The access of the external memory 1-6 is the same way as that in the native mode. An accessed instruction in the second type of software is transferred to the internal data bus 2-7 via the system data bus 1-4 and the data bus terminals 2-8. Then the instruction is applied to the conversion memory 2-3 as its address. Thus the instruction converted into one which can be directly executed by the CPU is applied to the instruction decoder 2-10 as described above. At this moment the address control circuit 2-4 operates as described above.

The last instruction of the emulation mode should include to change the state of the flip-flop of the emulation control circuit 2-15, if the first type of the software is to be executed after the execution of the second type of the software.

The initial state of the flip- flop of the emulation control circuit 2-15 may be determined to be a predetermined state for either the native mode or emulation mode by, for example, a power- on reset signal (not shown) provided within the processor 2-1. Thus, the instructions of the predetermined initial mode (native or emulation) should be first applied to the processor 2-1. Instead, the state of the flip-flop of the emulation control circuit 2-15 may be controlled externally, by providing the emulation control circuit 2-15 outside the processor 2-1 or by providing the processor 2-1 with a control terminal coupled to the emulation control circuit 2-15 and supplying a control signal to that terminal to determine or change the state of the emulation control circuit 2-15. In this case, either type of the software can be first supplied to the processor 2-1 by determining the state of the emulation control circuit 2-15 to be adapted for that type of the software.

According to the present invention, both the native mode and the emulation mode can be performed easily and at a high speed by the same CPU. Here, a high-speed instruction conversion can be performed by providing the conversion memory and the address control circuit on the same semiconductor chip as that on which the CPU is formed. However, the conversion memory and the address control circuit may be put in the outside of the CPU chip.

WHAT IS CLAIMED IS:

1. A microprocessor including a central processing unit adaptable for instructions of a first type and an instruction decoder for decoding the instructions of the first type comprising:

means for receiving an instruction of a second type unadaptable for said central processing unit;

memory means for preliminarily storing instructions of the first type, one of which corresponds to the received second type instruction in an operating function;

means for applying said second type instruction as at least one part of an address to said memory means;

means for reading the first type instruction out of said memory means according to the address; and

means for transferring the read-out first type instruction to said instruction decoder.

2. A microprocessor as claimed in claim 1, in which said applying means includes means for generating an address by which a first type instruction subsequent to said read-out instruction is successively designated.

3. A microprocessor as claimed in claim 2, in which said generating means generates said address according to a content of said read-out first type instruction.

4. A microprocessor as claimed in claim 1, in which said applying means applies two addresses to said memory means when the first type instruction corresponding to the received second type instruction consists of two bytes.

5. A microprocessor as claimed in claim 1, in which said memory means stores a specific information indicating

that the received second type instruction consists of a plurality of bytes.

6. A microprocessor as claimed in claim 5, in which said specific information is read out of said memory means according to a first byte of the received second type instruction.

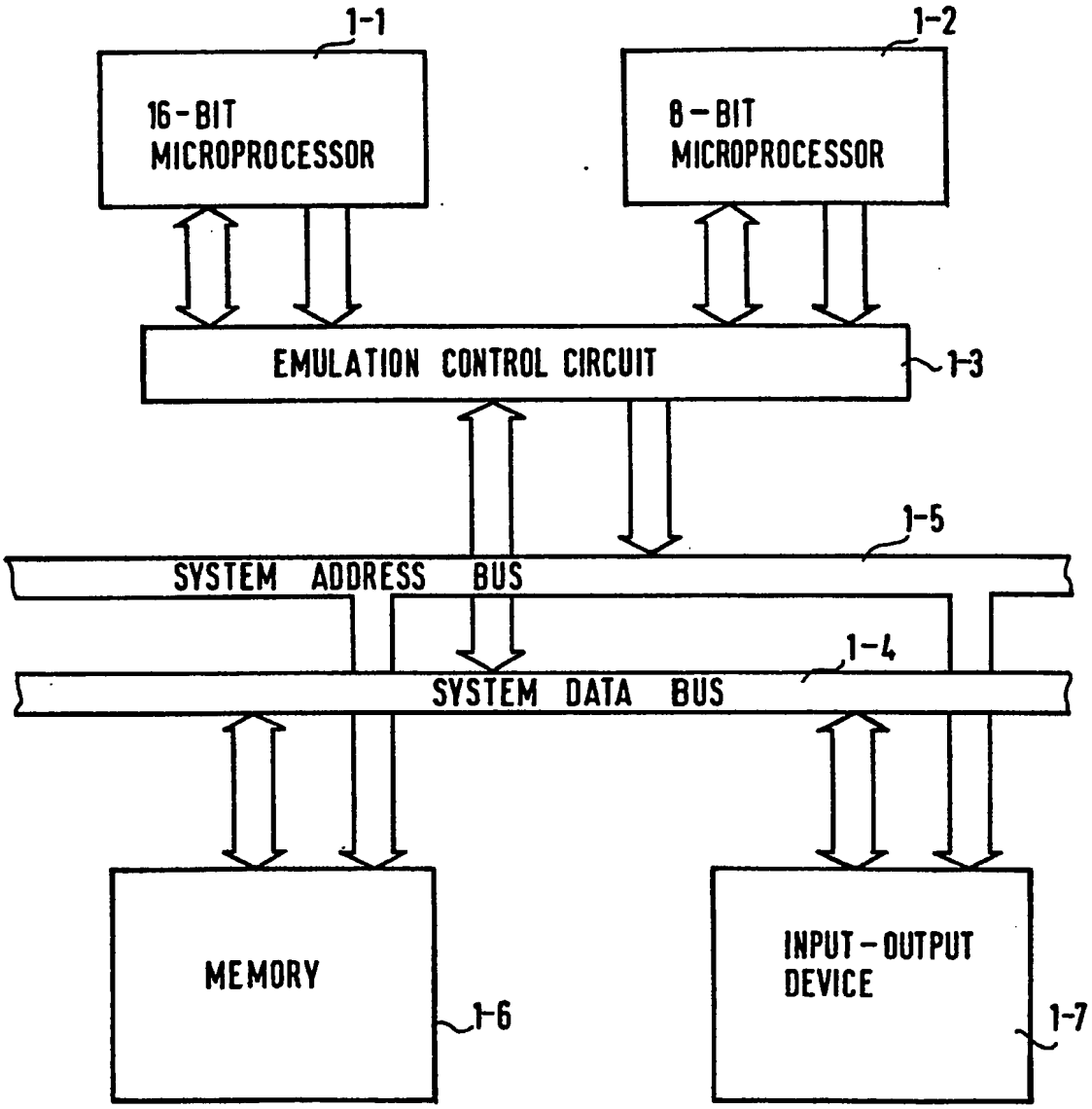
7. A microprocessor as claimed in claim 1, further comprising means for coupling said instruction decoder to either said receiving means or said memory means.

8. A microprocessor comprising an instruction decoder generating at least one control signal by decoding an instruction, an execution means executing a program according to the control signal generated from said instruction decoder, first means for receiving an instruction from the outside of said microprocessor, second means for applying the received instruction to said instruction decoder as it is, third means for inhibiting an application of said received instruction to said instruction decoder, fourth means for storing at least one instruction whose format is different from that of the received instruction, fifth means for generating an address accessing said fourth means according to the received instruction, and sixth means for transferring the instruction accessed by said fifth means to said instruction decoder.

9. A microprocessor as claimed in claim 8, in which the received instruction applied to said fifth means for generating the address of said fourth means is the instruction by which a correct execution of said execution means is disturbed.

10. A microprocessor as claimed in claim 8, in which said instruction decoder, said execution means and said first to sixth means are integrated on a single semiconductor chip.

FIG. 1


1-1
16-BIT MICROPROCESSOR
1-2
8-BIT MICROPROCESSOR
EMULATION CONTROL CIRCUIT
1-3
1-5
SYSTEM ADDRESS BUS
1-4
SYSTEM DATA BUS
MEMORY
1-6
INPUT-OUTPUT DEVICE
1-7

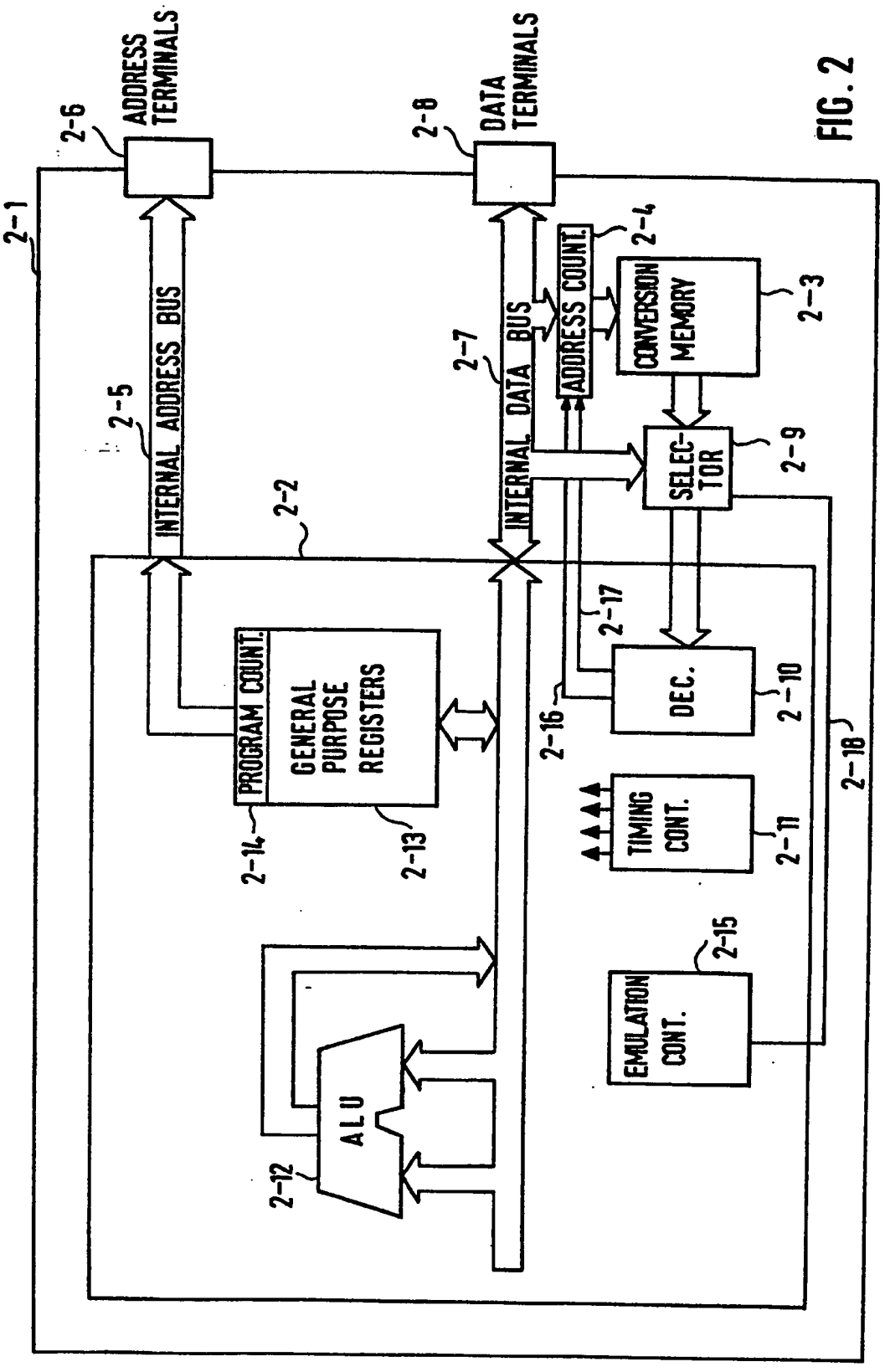

2-1
2-6
ADDRESS TERMINALS
2-8
DATA TERMINALS
2-5
INTERNAL ADDRESS BUS
2-7
INTERNAL DATA BUS
2-4
ADDRESS COUNT.
CONVERSION MEMORY
2-3
SELEC-TOR
2-9
2-2
2-17
2-16
DEC.
2-10
2-18
PROGRAM COUNT.
GENERAL PURPOSE REGISTERS
2-14
2-13
TIMING CONT.
2-11
2-15
EMULATION CONT.
ALU
2-12

FIG. 2

FIG. 3

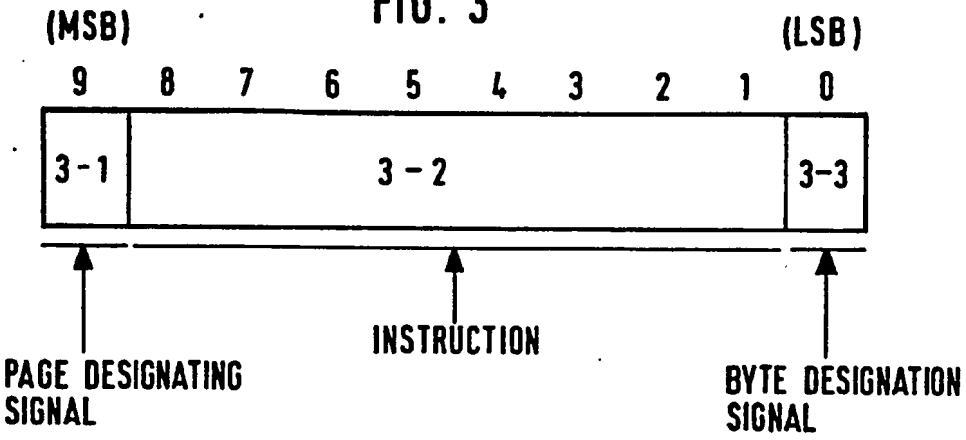

(MSB)
(LSB)
9 8 7 6 5 4 3 2 1 0
3-1
3-2
3-3
INSTRUCTION
PAGE DESIGNATING SIGNAL
BYTE DESIGNATION SIGNAL

FIG. 5

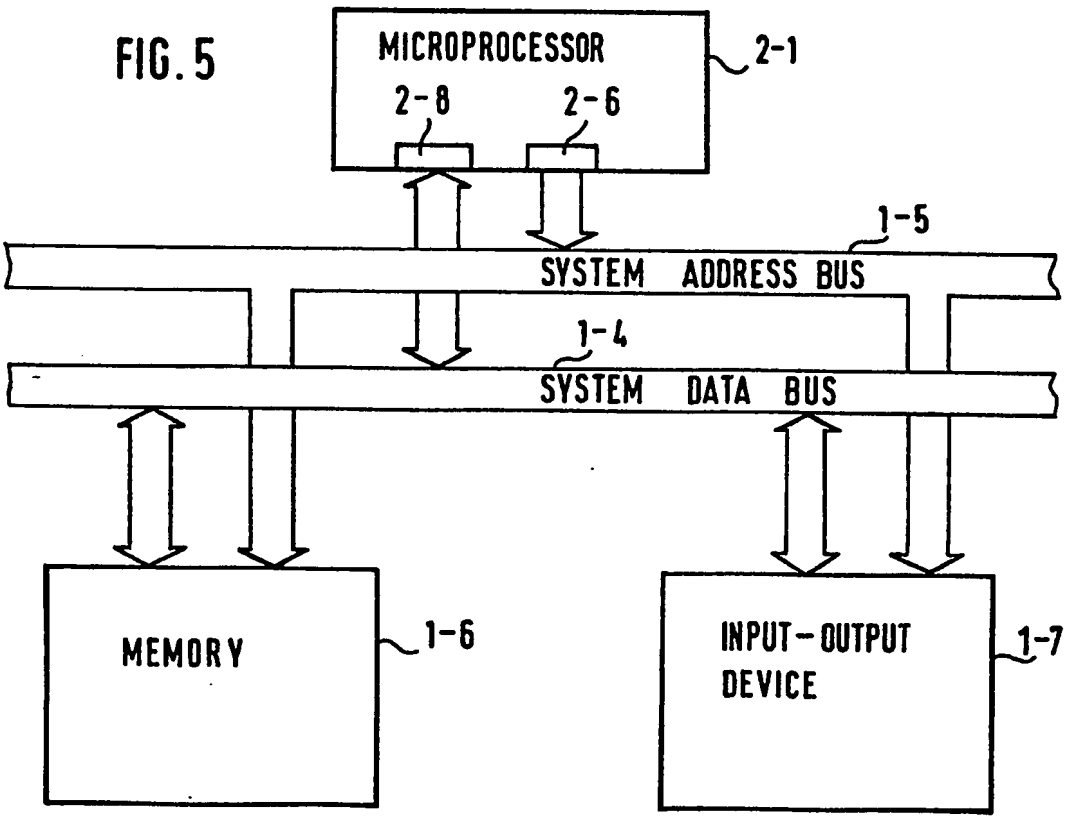
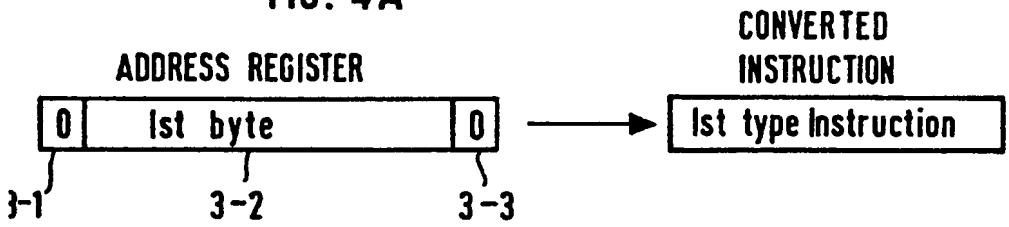

MICROPROCESSOR
2-1
2-8
2-6
1-5
SYSTEM ADDRESS BUS
1-4
SYSTEM DATA BUS
MEMORY
1-6
INPUT-OUTPUT DEVICE
1-7

FIG. 4A



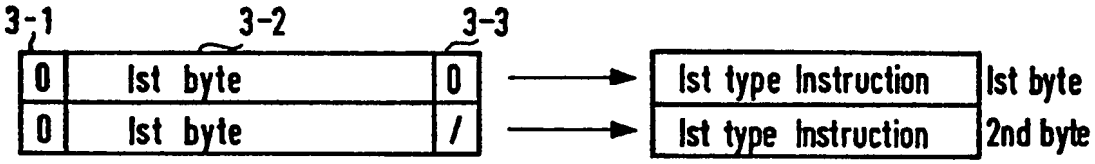
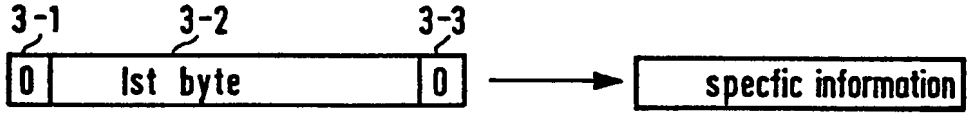


FIG. 4B



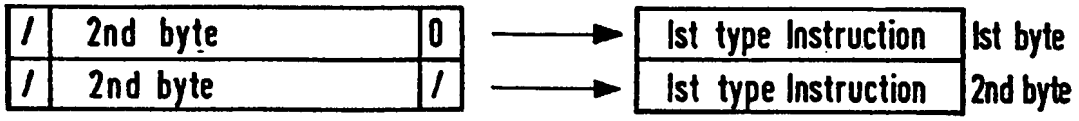


FIG. 4C